**Testing Strategy**

The testing strategy ensures all components function as expected under various scenarios:

1. **Unit Testing**: Verifies the correctness of individual methods, such as booking validation and file writing.
2. **Integration Testing**: Ensures proper interaction between the UI, logic, and data management layers.
3. **System Testing**: Evaluates the overall application to verify it meets functional and non-functional requirements.
4. **User Acceptance Testing (UAT)**: Confirms the system delivers a satisfactory user experience.

**Test Cases**

1. **Login Functionality**:
   * Input: Correct username and password.
   * Expected Output: Navigation to the movie selection page.
   * Result: Pass.
2. **Movie Selection**:
   * Input: Select "Bollywood Movie 1".
   * Expected Output: "Bollywood Movie 1" appears in the booking interface.
   * Result: Pass.
3. **Seat Booking**:
   * Input: Select seats "A1" and "A2".
   * Expected Output: Booking confirmation and updated booking file.
   * Result: Pass.
4. **Display Bookings**:
   * Input: Click "Display Bookings".
   * Expected Output: List of current bookings is displayed.
   * Result: Pass.
5. **Clear Bookings**:
   * Input: Click "Clear Bookings".
   * Expected Output: Booking data is cleared, and display area is empty.
   * Result: Pass.

**Performance Testing**

1. **Load Testing**:
   * Simulated multiple users booking seats simultaneously.
   * Result: Handled up to 50 users without noticeable delays.
2. **Response Time**:
   * Average response time for booking confirmation: **<2 seconds**.
   * Data display operations: **<1 second**.

**Issues Identified and Resolved**

1. **UI Highlighting Bug**: Fixed CSS styles to ensure selected seats were visually highlighted.
2. **Error Message Clarity**: Enhanced error messages for invalid inputs and operations.
3. **Data Synchronization**: Addressed delays in booking updates by optimizing data handling logic.